# Programming in MATLAB versus Python

## Introduction

MATLAB is a programming platform designed specifically for engineers and scientists. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics. MATLAB combines a desktop environment tuned for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly. The language, apps, and built-in math functions enable you to quickly explore multiple approaches to arrive at a solution. MATLAB lets you take your ideas from research to production by deploying to enterprise applications and embedded devices, as well as integrating with Simulink® and Model-Based Design. MATLAB platform is a proprietary product acquired at substantial cost from MathWorks, Inc MATLAB provided a powerful learning environment for students in science and engineering. They can construct solutions to problems, confirm and revise their solutions rapidly in MATLAB and look at the solutions visually in 2D, 3D, and even higher dimensions as needed.

Python is the foremost general-purpose programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python supports libraries modules and packages, adding power to the standard python distribution package. It allows for code modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. The edit-test-debug cycle is relatively is convenient and fast with open-source free distribution versions such as Anaconda. Because Python is open and free, it is very easy for everyone to design packages or other software tools further that extending the Python program development environment.

Much of the MATLAB functionality can be achieved in Python with Python packages such NumPy, SciPy and Matplotlib. However, the model-based design feature of MATLAB Simulink with graphical design and modeling environment is not available for Python at this time. MATLAB comes in a robust package as a commercial product. Python on the other hand is influenced constantly by changes made available instantly leading sometime to code incompatibility of new versions versus older versions.

The following are some side-by-side examples of programming in MATLAB and Python.

**Section 1: Simultaneous Equations and Matrices**
Consider the following two simultaneous equation in variable x and y.
$x-y=40$ (1)
$x-2y=-66$ (2)
The solution needed is for values of x and y that will satisfy the two equations simultaneously.

We could find the solution by direct substitution.
From equation (1), x=40+y; substituting it for x in (2), we get
40+y-2y=-66; Therefore -y=-106 i.e. y=106
Now substituting this value (1), we get x-106=40 i.e. x=146
Solution: x=146 y=106

A shortened notation of denoting set of simultaneous linear algebraic equations is Ax=b where A is the matrix of coefficients of the simultaneous equation, x denotes variable in the form of $x_{11}$, $x_{12}$, …, $x_{1n}$ for the first equation; $x_{21}$, $x_{22}$, …, $x_{2n}$ for the second equation, and so on. And b is the column vector of constants representing the right-hand side of the equations. The equations are assumed to be well conditioned i.e. they lead to a meaningful solution. If the matrix is named A, its inverse is denoted as A'.

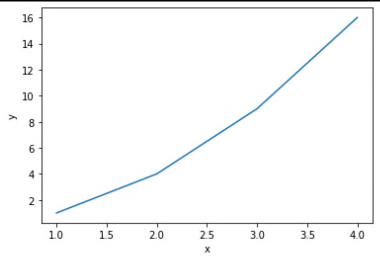
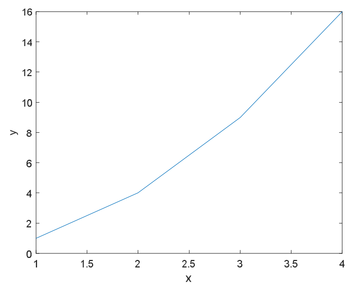
**Comparison of MATLAB and Python Solutions of Simultaneous Equations**

**Example 1:**

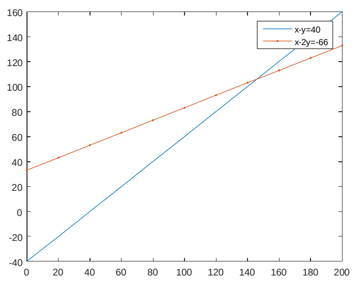
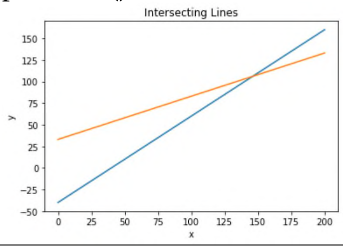
| # | MATLAB | # | Python |
|---|---|---|---|
| 1 | `>> a= [1 -1; 1 -2]` | 1 | `import numpy as np` |
| 2 | `a =` | 2 | `A= np.array([[1, -1], [1, -2]])` |
| 3 | `   1    -1` | 3 | `C= [40, -66]` |
| 4 | `   1    -2` | 4 | `s= np.linalg.solve(A,C)` |
| 5 | `c =` | 5 | `s` |
| 6 | `   40` | 6 | `array([146., 106.])` |
| 7 | `  -66` | 7 | (Alternatively) |
| 8 | `s= a\c` | 8 | `Ainv= np.linalg.inv(A)` |
| 9 | `s =` | 9 | `s= np.dot(Ainv,C)` |
| 10 | `  146` | 10 | `s` |
| 11 | `  106` | 11 | `array([146., 106.])` |
| | (Note: MATLAB provides a natural way of describing operation for problems in science and engineering making use of matrix operations. >> is prompt from the system for stating desired operations in a console session where statements are interpreted and executed as they are entered.) | | (Note: The first line imports Package numpy for numerical methods and names it as np. The second and thirst line are matrix of coefficients and the corresponding array for the right side of the equation. The fourth line call np methods for solution of simultaneous algebraic equation, and final line show the solution in the form of values for x and y.) |

**Example 2: Simple Plots**

| MATLAB | Python |
|---|---|
| `>> plot([1, 2, 3, 4], [1, 4, 9, 16]), xlabel('x'), ylabel('y')` | `import matplotlib.pyplot as plt`<br>`plt.plot([1, 2, 3, 4], [1, 4, 9, 16])`<br>`plt.xlabel('x')`<br>`plt.ylabel('y')`<br>`plt.show()` |

| | |
|---|---|
|  <br>Note: Basic plotting functions are already and need not be imported or loaded. |  |

**Example 3: Multiple Functions on the Same Plot**

| MATLAB | Python |
|---|---|
| >> x= [0: 20: 200]<br>>> y1= x-40<br>>> y2=x/2 + 33<br>>> plot(x,y1, x, y2,'.-'), legend('x-y=40', 'x-2y=-66')<br> <br>(Note: MATLAB has tools for plotting in, generally more powerful and easier to work with as compared to Python.)<br>(The lines intersect at x=146 and y= 106 as expected) | import numpy as np<br>import matplotlib.pyplot as plt<br>x=np.linspace(0.0, 200.0, num=20)<br>t= x[:20]<br>y1=t-40<br>plt.plot(t,y1, ‘b’)<br>y2=t/2 + 33<br>plt.plot(t,y2, ‘r’)<br>plt.xlabel('x')<br>plt.ylabel('y')<br>plt.title('Intersecting Lines')<br>plt.show()<br> |

## Session 2: A Simple Example of using a Numerical Method

**Example 4: Finding a Square Root of a Number**

MATLAB

```
% Calculate square root of a given number by
% Newton's method
% Read input
x=input('Please type a number for calculating
square root:');
if x<=0
display('The number must be positive'); end;
% Initial guess.
xstart = x/2;
% Iteration loop to compute square root.
for i = 1:100
xnew = (xstart + x/xstart)/2;    % new estimate
of square root.
display(xnew);% print xnew.
if abs(xnew - xstart)/xnew < eps
        break % on convergence.
end;
xstart = xnew;  % update estimate of square
root.
end
```

**Running this program In MATLAB**

```
>> Newton
% MATLAB response
Please type a number for calculating square
root:20
xnew =  6
xnew =  4.6667
xnew =  4.4762
xnew =  4.4721
xnew =  4.4721
xnew =  4.4721
```

Python

```
import sys       # to fetch epsilon value
eps = sys.float_info.epsilon
# Calculate square root of a given number by
# Newton's method
# Read input
x= input("Please type a number for root finding
root:\n")
x= float (x)
if x<=0:
   print('The number must be positive')
xstart = x/2 # initial guess
max= 1
while max < 5:
   xnew=(xstart + x/xstart)/2 #new estimate
   print(xnew)
   if abs(xnew-xstart)/xnew < eps:
      break
   else:
      max= max + 1
      xstart=xnew
print (xnew)

(Run)
Please type a number for root finding root:
20
6.0
4.666666666666667
4.476190476190476
4.472137791286727
4.472137791286727
```

**Session 3: Object-Oriented Programming**

Object-oriented programming (OOP) is a computer programming framework that organizes solutions to problems in terms of objects in the world, their attributes, and behavior. Consider a bank that deals with customers. Customer records are maintained in terms of a set of well-defined attributes, also known as properties or fields. Each customer is permitted a well-defined set of operations, also known as the methods that may applied for those operations, visualized as permissible behavior for each instance of customer account. A customer or his account may be viewed as a class. The instances of this class, also called objects, are the specific accounts. Each account is an object with attributes and behavior common to the customer account class. The following example, defines an Investment as a class and each investment is an object or class instance. The key feature of object-oriented programming is called encapsulation. Each object has the same attributes defined in the class and only the operations defined for the class can be performed on an instance of a class. Those performing the operations need not be aware of how an operation is performed and the methods for those operations can modified without making the users aware of it.

**Example 5. Object-Oriented with Classes, Class Properties (Attributes) and Class Methods (Behaviors)**

MATLAB

```
classdef Investment
   properties
      Amount
   end
   methods
      function obj = Investment(val)
         if nargin == 1
            obj.Amount = val;
         end
      end
      function r = earning(obj,rate)
         r = [obj.Amount] * rate/100;
      end
      function r = cumulate(obj, rate, years)
         r = obj.Amount*(1+rate/100)^years;
      end
   end
end
```

```
>> inv1=Investment
inv1 =
Investment with properties:
   Amount: []
>> inv1.Amount=500.6789
>> inv1.earning(3)
ans =  15.0204
>> inv1.cumulate(3,1)
ans = 515.6993
>> inv1.cumulate(3,2)
ans = 531.1702
```

Python

```
class Investment:
   def __init__(self, a=0):
      self.amount=a  #amount is an instance atribute of the class

   def plus(self, x):
      self.amount += x

   def minus(self, x):
      self.amount -= x

   def yearlyGrowth(self, x):
      self.amount *= 1+x/100   # x% rate

   def cumulate (self, x, y):
      self.amount *= (1+x/100)**y # x% growth for y years
```

```
inv1=Investment(500.6789)
inv1.yearlyGrowth(3)
inv1.amount
515.6992
inv1.plus(50)
inv1.amount
565.6992
inv1.minus(65)
inv1.amount
500.6992
inv1.cumulate(3,2)
inv1.amount
531.17024501
```

The above examples are intended to provide a flavor of some problems with side-by-side coding in MATLAB versus Python, and not intended to teach either MATLAB or Python, both beyond the scope of this short note.